4. TESTES PARA AS SEQÜÊNCIAS DE NÚMEROS PSEUDO-RANDÔMICOS

4.1. INTRODUÇÃO

É indispensável testar as seqüências de números pseudo-randômicos para saber se elas são apropriadas para os nossos estudos. Isto se deve porque as seqüências de números geradas pelo computador não é randômica, mas sim pseudo-randômica, algumas seqüências possuem propriedades diferentes de outras, e por isso devemos tomar o cuidado de estudar as seqüências antes de utizá-las em qualquer aplicação.

O usuário de uma seqüência de números pseudo-randômicos pode projetar os seus próprios testes, de acordo com a propriedade que queira dar às seqüências. Na maioria das vezes desejamos sequências de números randômicos uniformemente distribuídas, assim apresentaremos alguns testes, como o teste χ^2 , para a obtenção dessa propriedade desejada.

Podemos classificar os testes para as seqüências de números pseudo-randômicos em duas categorias básicas. A primeira categoria é a chamada categoria dos testes empíricos que são os testes realizados computacionalmente através de programas de computador e a segunda categoria é a chamada categoria dos testes teóricos, que não será tratada aqui.

Nos testes precisaremos da seqüência de números aleatórios v_n onde

$$v_n = v_0, v_1, v_2, \ldots$$

e cada v_n é um número real entre zero e um. Alguns testes usarão outras seqüências, do tipo inteiro por exemplo, que são definidas por y_n onde

$$y_n = y_0, y_1, y_2, \ldots$$

e estas seqüências de inteiros são seqüências modificadas que são definidas pela regra abaixo:

$$y_n = \text{floor}(d\, v_n) .$$

A seqüência acima terá uma distribuição uniforme no intervalo [0 , d-1].

Dentre os testes do tipo empírico, estudaremos os testes de

- *equidistribuição ou teste de freqüência,*
- *teste serial,*
- *teste de intervalo,(abertura)*
- *teste de permutações,*
- *teste de rodagem,*
- *teste do máximo.*

Além dos testes acima existem dois importantes testes que estudaremos agora, são os os testes estatísticos χ^2 e Kolmogorov-Smirnov de uniformidade.

4.2. TESTES ESTATÍSTICOS

4.2.1. GERADOR DE NÚMEROS PSEUDO-ALEATÓRIOS

O gerador de números pseudo-randômicos abaixo foi criado partir do método de congruência linear e será utilizado para os exemplos dos testes estatísticos e empíricos.

Para usar o gerador, nós devemos inicializar a semente, caso contrário o gerador tomará como semente o valor 1739. Para inicializar a semente nós fazemos:

_semente := valor ;

Onde
valor é um inteiro positivo entre zero e o módulo do gerador.

Sintaxe do gerador:

random ( )

o prodecimento random retorna números no intervalo [0 , 1[

```
> random:=proc()
> local _a,_c,_m;global _semente;
> _a:=47061;
> _c:=34999787;
> _m:=10^9;
> if nargs>0 then ERROR(`Você não deve digitar argumentos`) fi;
> if not assigned(_semente) then _semente:=1739; fi;
```

```
> _semente:=(_semente*(_a)+_c)mod(_m);
> RETURN(evalf(_semente/_m,16));
> end:
>
```

4.2.2. TESTE ESTATÍSTICO χ^2

CONCEITO

O teste χ^2 de uniformidade foi introduzido por Karl Pearson em 1900. É o teste estatístico mais conhecido e é utilizado em conjunto com outros testes.

Para explicar o teste vamos considerar como exemplo o lançamento de um dado. Vejamos a tabela de probabilidade abaixo:

Valor de s	1	2	3	4	5	6
Probabilidade ps	1/3	1/3	1/3	1/3	1/3	1/3

onde s é o número que o dado fornece ao ser lançado e $p\,s$ é a probabilidade de cair um dos 6 números do dado. Se lançarmos o dado n vezes, obteremos os valores de s aproximadamente $n\,p_s$ vezes, em média. Por exemplo, em 18 lançamentos, observamos:

Valor de s	1	2	3	4	5	6
Número observado	3	1	0	5	5	4
Número esperado $n\,ps$	3	3	3	3	3	3

Veja que o número observado, neste exemplo, é diferente do número esperado na maioria dos casos. Em vista da seqüência acima, poderíamos perguntar: Quando ou não um dado estaria viciado? Podemos responder essa pergunta *probabilisticamente*. Nós podemos através da estatística dizer quão provável ou improvável certos eventos são.

Uma forma de realizar a estatística é tomar o quadrado das diferenças entre o valor observado e o valor esperado e depois podemos tomar as somas e calcular V:

$$V = (Y_1 - n\,p_1)^2 + (y_2 - n\,p_2)^2 + \ldots + (y_6 - n\,p_6)^2 \qquad (1)$$

De certo ponto de vista uma seqüência de números pseudo-randômicos é ruim quando retorna um valor relativamente alto de V. Agora vamos considerar a divisão de cada termo de (1) por $n\,p_i$ e teremos:

$$V = \frac{(Y_1 - n\,p_1)^2}{n\,p_1} + \frac{(y_2 - n\,p_2)^2}{n\,p_2} + \ldots + \frac{(y_6 - n\,p_6)^2}{n\,p_6}$$

Esta é a chamada estatística χ^2 dos valores observados y_s neste experimento de lançamento de dado.

Generalizando, supomos que todas as observações caem em k categorias, nós fazemos n observações independentes e formamos a estatística

$$V = \sum_{s=1}^{k} \frac{(y_s - n\,p_s)^2}{n\,p_s} \qquad \text{(estatística } \chi^2\text{)}$$

Pela expanção de $(y_s - n\,p_s)^2$ e usando o fato que
$y_1 + y_2 + ... + y_k = n$ e $p_1 + p_2 + ... + p_k = 1$ nós chegamos à fórmula

$$V = \frac{\sum_{s=1}^{k} \frac{Y_s^{\,2}}{p_s}}{n} - n$$

que frequentemente facilita o cálculo da estatística χ^2 . Agora podemos perguntar: O que constitui um valor razoavelmente bom para V tal que a seqüência seja uniformemente distribuída? Para responder a essa pergunta nós comparamos o valor da estatística V obtida, com uma *tabela de valores:* **Tabela 1**.
A Tabela 1 dá a distribuição χ^2 *com v graus de liberdade* para vários valores de v.

A linha da tabela com $v = k - 1$ é usada para a verificação, onde v é o grau de liberdade e k é o numero de classes. O grau de liberdade é um a menos que o número de classes.

Para resumir, contamos o número de observações que caem em cada categoria (classe) e calculamos a estatística $V = \chi^2$. Então V é comparada com a Tabela 1, onde $v = k - 1$. Se V é maior que 99% ou menor que 1% nós regeitamos a seqüência pois ela é **não suficientemente pseudo-randômica.** Se V cai entre 99% e 95% ou entre 5% e 1%, a seqüência é considerada como **suspeita de ser pseudo-randômica.** Se V cai entre 95% e 90% ou 10% e 5% a seqüência é considerada como **quase suspeita de ser pseudo-randômica.** O teste χ^2 é freqüentemente feito ao menos três vezes sobre diferentes conjuntos de dados, se ao menos dois daqueles **3 resultados são suspeitos** a seqüência é considerada **não suficientemente randômica.**

TABELA 1:

	P=99%	p=95%	p=75%	p=50%	p=25%	p=5%	p=1%
v = 01	0,00016	0,00393	0,1015	0,4549	1,323	3,841	6,635
v = 02	0,02010	0,1026	0,5753	1,386	2,773	5,991	9,210
v = 03	0,1148	0,3518	1,213	2,366	4,108	7,815	11,34
v = 04	0,2971	0,7107	1,923	3,357	5,385	9,488	13,28
v = 05	0,5543	1,1455	2,675	4,351	6,626	11,07	15,09
v = 06	0,8720	1,635	3,455	5,348	7,841	12,59	16,81
v = 07	1,239	2,167	4,255	6,346	9,037	14,07	18,48
v = 08	1,646	2,733	5,071	7,344	10,22	15,51	20,09
v = 09	2,088	3,325	5,899	8,343	11,39	16,92	21,67
v = 10	2,558	3,940	6,737	9,342	12,55	18,31	23,21
v = 11	3,053	4,575	7,584	10,34	13,70	19,68	24,73
v = 12	3,571	5,226	8,438	11,34	14,84	21,03	26,22
v = 15	5,229	7,261	11,04	14,34	18,25	25,00	30,58
v = 20	8,260	10,85	15,45	19,34	23,83	31,41	37,57
v = 30	14,95	18,49	24,48	29,34	34,80	43,77	50,89
v = 50	29,71	34,76	42,94	49,33	54,33	67,50	76,15

No teste χ^2 somente o valor de $v = k - 1$ afeta no resultado, então ao consultarmos a tabela de valores devemos observar as linhas em que é dado o valor de v para o teste.

As tabelas de valores para os testes são aproximadas e as aproximações são válidas para valores satisfatoriamente grandes de n. Tais valores devem ser escolhidos de tal forma que o valor esperado $n\,p_s$ seja maior ou igual a cinco. Se esta regra for seguida, teremos um teste mais acurado.

O procedimento abaixo verifica se uma seqüência de números pseudo-randômicos está uniformemente distribuida entre os n subintervalos do intervalo [0,1[. Este teste divide o intervalo [0,1[em n subintervalos chamado classes que não se sobrepõem e conta quantos números da seqüência caem em cada intervalo, formando assim uma nova seqüência, e uma estatística χ^2 mede se a seqüência é uniformemente distribuída.

Sintaxe:

chi (v, s, n)

- v é o nome da seqüencia de números pseudo-randômicos;
- s é tamanho da seqüência.
- n é o número de classes.

```
> chi:=proc(v::string,s::posint,n::posint)
> local i,j,b,w,esp:
> b:=1/n:
> if nargs>3 then ERROR(`Você deve digitar no máximo 3
  argumentos`) fi:
> for i from 0 to n do w[i]:=0: od:
> for j from 0 to n-1 do
> for i from 0 to s-1 do
```

```
> if (v[i]>=j*b) and (v[i]<(j+1)*b) then
> w[j]:=w[j]+1: fi:
> od:
> od:
> esp:=s/n:
> RETURN(evalf(sum((w['i']-esp)^2/esp,'i'=0..n-1))):
> end:
>
```

EXEMPLOS

Os exemplos abaixo utilizam o gerador de números pseudo-randômicos que foi definido neste capítulo.

Os exemplos abaixo testam seqüências de números entre [0,1[.

```
> _semente:=6097:
```
escolhendo a semente do gerador

```
> for i from 0 to 200 do
> w[i]:=random():
> od:
```
definindo a seqüência *w*

```
> chi(w,200,10);
```
com 9 graus de liberdade passa

6.200000000

```
> chi(w,200,5);
```
com 4 graus de liberdade passa

4.

```
> _semente:=100:

> for i from 0 to 200 do
> w[i]:=random();
> od:

> chi(w,200,10);
```
com 9 graus de liberdade não passa

2.800000000

```
> chi(w,200,5);
```
com 4 graus de liberdade não passa

2.300000000

```
>
```

4.2.3. TESTE ESTATÍSTICO KOLMOGOROV-SMIRNOV

CONCEITO

O teste Kolmogorov-Smirnov (KS) é um teste estatístico como o teste χ^2 que avalia se uma quantidade pseudo-randômica X está uniformentente distribuída no intervalo [0,1[.

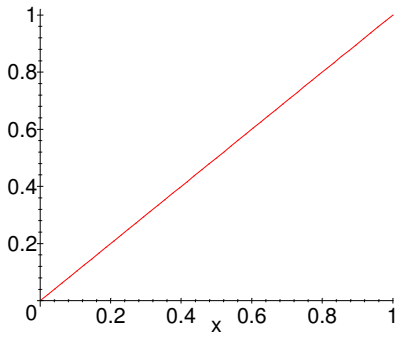
Para realizar o teste sobre quantidades finitas ou infinitas nós usamos uma função de distribuição f(x). Essa função é definida abaixo:

f(x) = *probabilidade que* ($X \le x$)

```
> f:=x->x:
```

FUNÇÃO DE DISTRIBUIÇÃO

```
> plot(f(x),x=0..1);
```



O gráfico acima é o da função de distribuição de um número pseudo-randômico real unformemente distribuído entre zero e um. É fácil deduzir que para uma quantidade $X \le x$ tem função de distribuição f(x) = x para uma distribuição uniforme.

Se nós fizermos n observações independentes de uma quantidade pseudo-randômica X, nós podemos formar uma *função de distribuição empírica* $f_n(x)$, onde

$f_n(x)$ = (*numero de* $X_1, X_2, \ldots, X_n$)/n

O teste KS é baseado na diferença entre f(x) e $f_n(x)$. Uma má seqüência dará uma função de distribuição empírica que não se aproxima suficientemente bem de f(x).

Nós realizamos o teste com as seguintes estatísticas:

$K_n+ = \sqrt{n}$ max ($f_n(x)$ − f(x), $x = -\infty .. \infty$);

$K_n- = \sqrt{n}$ max (f(x) − $f_n(x)$, $x = -\infty .. \infty$);

Onde K_n+ mede o maior desvio quando f_n é maior que f e K_n- mede o maior desvio quando f_n é menor que f.

Agora com os valores K_n+ e K_n- observamos a tabela de porcentagens (**Tabela 2**) para determinar se a seqüência é suficientemente randômica.

TABELA 2:

	p=99%	p=95%	p=75%	p=50%	p=25%	p=5%	p=1%
v = 01	0.01000	0.05000	0.2500	0.5000	0.7500	0.9500	0.9900
v = 02	0.01400	0.06749	0.2929	0.5176	0.7071	1.0980	1.2728
v = 03	0.01699	0.07919	0.3112	0.5147	0.7539	1.1017	1.3589
v = 04	0.01943	0.08789	0.3202	0.5110	0.7642	1.1304	1.3777
v = 05	0.02152	0.09471	0.3249	0.5245	0.7674	1.1392	1.4024
v = 06	0.02336	0.1002	0.3272	0.5319	0.7703	1.1463	1.4144
v = 07	0.02501	0.1048	0.3280	0.5364	0.7755	1.1537	1.4246
v = 08	0.02650	0.1086	0.3280	0.5392	0.7797	1.1586	1.4327
v = 09	0.02786	0.1119	0.3274	0.5411	0.7825	1.1624	1.4388
v = 10	0.02912	0.1147	0.3297	0.5426	0.7845	1.1658	1.4440
v = 11	0.03028	0.1172	0.3330	0.5439	0.7863	1.1688	1.4484
v = 12	0.03137	0.1193	0.3357	0.5453	0.7880	1.1714	1.4521
v = 15	0.03424	0.1244	0.3412	0.5500	0.7926	1.1773	1.4606
v = 20	0.03807	0.1298	0.3461	0.5547	0.7975	1.1839	1.4698
v = 30	0.04354	0.1351	0.3509	0.5605	0.8036	1.1916	1.4801

Uma propriedade que o teste KS apresenta é que nós podemos dividir a seqüência em *n* subseqüências e aplicar KS em cada subseqüência formando assim uma nova seqüência e então aplicar KS nessa seqüência, isto é, vejamos um exemplo:

Supomos que nós temos a seguinte seqüência de 20 números pseudo-randômicos:

1, 5, 19, 14, 2, 0, 15, 11, 8, 7, 18, 3, 9, 17, 4, 16, 6, 10, 12, 13

Agora nós vamos dividir essa seqüência em 2 subseqüências com dez números cada e teremos:

1, 5, 19, 14, 2, 0, 15, 11, 8, 7 (1)
18, 3, 9, 17, 4, 16, 6, 10, 12, 13 (2)

Nós aplicamos o teste KS à seqüência (1) e à seqüência (2) e obteremos dois números k[1] e k[2]. À seqüência k, nós aplicamos novamente o teste KS.

O interessante é que todo esse procedimento é o mesmo que se nós aplicassemos o teste KS à primeira seqüência de vinte números. O que isso nos ajuda é que com esse procedimento nós podemos saber o resultado do teste KS para um grau de liberdade reduzido, facilitando o uso da **Tabela 2**. O procedimento também possibilita a detecção se o gerador apresenta uma boa randomicidade local.

O teste Kolmogorov-Smirnov pode ser aplicado em conjunto com o teste χ^2 e isso consiste no método *ad hoc* . Supomos que fazemos *n* testes χ^2 sobre diferentes partes de uma quantidade aleatória *X*, e os valores v_1, v_2, ... , v_n foram obtidos. Aplicando as estatísticas K_n+ e K_n- na seqüência v_1, v_2, ... ,v_n podemos saber se ela é suficientemente randômica.

Uma importante diferença entre o teste KS e o teste χ^2 é que o teste KS é aplicado sobre a distribuição f(*x*) *contínua*, enquanto o teste χ^2 é aplicado sobre *distribuições não-contínuas*,

isto é, que apresenta saltos.

O programa abaixo calcula qual é o valor máximo dentro de um conjunto de números de uma seqüência. Ele também calcula onde está localizado o valor máximo dentro da seqüência que é dado pela variável global *pit*.

```
> maximo:=proc(x::string,i::integer,j::integer)
>  local aux,mxim,l; global pit;
>    aux:=-(10^12);
>     for l from i to j do
>      if x[l]>aux then
>       aux:=x[l]:
>       pit:=l:
>      fi:
>     od:
>  RETURN(aux):
> end:
>
```

O programa abaixo coloca uma seqüência em ordem crescente para o teste KS.

```
> crescente:=proc(x::string,i::integer,j::integer)
>  local z,l,a; global s;
>   z:=j:
>   s:=array(i..j);
>   for l from i to j do
>    s[l]:=x[l]:
>   od:
>    while z>i do
>     for l from i to z do
>      a[l]:=s[l]:
>     od:
>      s[z]:=maximo(s,i,z):
>      s[pit]:=a[z]:
>      z:=z-1:
>    od:
>    RETURN(evalm(s));
> end:
>
```

PROGRAMA KOLMOGOROV-SMIRNOV

O procedimento abaixo usa os procedimentos definidos acima para calcular o valor da estatística KS.

Sintaxe:

KS (v, w, h)

onde v = nome da seqüência pseudo-randômica
w = tamanho da seqüência de números pseudo-aleatórios
h = variável que retorna o teste K+ ou K-

```
> KS:=proc(v::string,w::posint,h)
>  local j,f,kpos,kneg;global r,z;
>
>    crescente(v,0,w-1):
>     f:=x->x:   função de distribuição
>      for j from 0 to w-1 do
>       r[j]:=(j+1)/w-f(s[j]):
>       z[j]:=f(s[j])-(j)/(w):
>      od:
>       if h=`k+`then
  kpos:=sqrt(w)*maximo(r,0,w-1);
  RETURN(evalf(kpos)); fi;
>       if h=`k-`then kneg:=sqrt(w)*maximo(z,0,w-1);
>            RETURN(evalf(kneg)); fi;
> end:
>
```

EXEMPLOS

O exemplo abaixo é a resposta do exercício 5 da seção 3.3.1 do livro Seminumerical Algorithms de D. E. Knuth.

Para resolver, nós criamos a seqüência x com os números dados e depois calculamos o valor de K+ e K- através do nosso programa.

```
> x[0]:=0.414:          x[10]:=0.442:
> x[1]:=0.732:          x[11]:=0.434:
> x[2]:=0.236:          x[12]:=0.141:
> x[3]:=0.162:          x[13]:=0.017:
> x[4]:=0.259:          x[14]:=0.318:
> x[5]:=0.442:          x[15]:=0.869:
> x[6]:=0.189:          x[16]:=0.772:
> x[7]:=0.693:          x[17]:=0.678:
> x[8]:=0.098:          x[18]:=0.354:
> x[9]:=0.302:          x[19]:=0.718:

> KS(x,20,`k+`);
                              1.153811077
> KS(x,20,`k-`);
```

.2146625259

Pela tabela 2, temos uma seqüência satisfatória de números.

Vamos definir agora uma seqüência com 200 números usando um gerador e testá-la:

```
> _semente:=6097:

> for i from 0 to 200 do
> F[i]:=random():
> od:

> KS(F,200,`k+`);
                                .7918069728
> KS(F,200,`k-`);
                                .1808648897
>
```

4.3. TESTES EMPÍRICOS PARA AS SEQÜÊNCIAS DE NÚMEROS PSEUDO-RANDÔMICOS

4.3.1. TESTE DE EQUIDISTRIBUIÇÃO

4.3.1.1. CONCEITO

Uma seqüência deve ser uniformente distribuída entre zero e um e para verificar isso nós usamos o teste Kolmogorov-Smirnov ou o programa que foi criado abaixo.

O teste de equidistribuição é similar ao teste χ^2, como característica verifica se as seqüências de números inteiros no intervalo [0, d [está uniformemente distribuída. Para fazer isso, nós construímos uma seqüência de números inteiros tomando-se o menor inteiro maior ou iqual a $d\,U$ onde U é uma seqüência de números randômicos no intervalo [0, 1 [. Depois contamos a quantidade de números que caem em cada classe 1, 2, ..., $d-1$ e realizamos então uma estatística χ^2 sobre os resultados observados.

É de se esperar que se uma seqüência U de números randômicos esteja uniformemente distribuída no intervalo [0, 1 [, então ela também estará num outro intervalo qualquer, gerado a partir de U.

4.3.1.2. O PROGRAMA

O programa que segue realiza o teste de equidistribuição.
O comando ***equidis*** recebe três argumentos.
$k = d - 1$ graus de liberdade.

Sintaxe:

equidis (v, w, d)

onde *v* é o nome da seqüência
w é o tamanho da seqüência
d é o fator que definirá uma seqüência de inteiros entre [0 , d [

```
> equidis:=proc(v,w::posint,d::posint)
>  local i,y,r,j,esp,ob:
>    for i from 1 to w do
>     ob[i-1]:=0;
>     y[i]:= floor(d*v[i]);
>    od:
>     for r from 0 to d-1 do
>      for j from 1 to w do
>       if y[j]=r then
>        ob[r]:=ob[r]+1:
>       fi:
>      od:
>     od:
>      esp:=w/d:
>      RETURN(evalf(sum((esp-ob[k])^2/esp,k=0..d-1)));
> end:
>
```

4.3.1.3. EXEMPLOS

Vamos testar se a sequência abaixo satisfaz o teste de equidistribuição.

```
> _semente:=7750:
> for i from 1 to 10000 do
> x[i]:=random():
> od:

> equidis(x,10000,4);
```

2.405600000

```
> equidis(x,10000,3);
```

3.762200000

Observando-se a tabela de valores do teste χ^2 podemos verificar se a seqüência acima satisfaz o teste de equidistribuição.

```
>
```

4.3.2. TESTE DE ABERTURA

4.3.2.1. CONCEITO

O teste de abertura mede se os comprimentos das aberturas que aparecem num certo sub-intervalo do intervalo [0 , 1] da seqüência V_n está uniformemente distribuído, isto é, ele verifica se num dado intervalo entre zero e um, a quantidade de números concecutivos que caem nesse intervalo está uniformemente distribuída; α e β são dois números tais que $0 \le \alpha$ e $\alpha < \beta$ e $\beta \le 1$, o comprimento das subseqüências v_j, v_{j+1},...,v_{j+r} no qual v_{j+r} cai entre α e β e os outros v's que não caem é chamado de abertura de comprimento r. O intervalo [α , β] é o sub-intervalo do intervalo [0 , 1].

O teste consiste de verificar quantos números sucessivos caem dentro do intervalo [α , β]. A quantidade de números sucessivos que caem dentro desse intervalo é o comprimento de abertura. Então esses comprimentos obtidos são verificados com um teste χ^2 para se concluir se são suficientemente randômicos.

4.3.2.2. PROGRAMA

$k = t$ graus de liberdade.

Sintaxe:

abertura (v, w, t, alpha, beta)

onde *v* é o nome da seqüência de números
w é o comprimento da seqüência
t é o número de classes
[alpha, beta] é o intervalo verificado

```
> abertura:=proc(v::string,w::posint,t::posint,alpha,beta)
>  local s,q,r,p,k,i:global j,count:
>
>    j:=-1;
>     s:=0;
>      for r from 0 to t do
>       count[r]:=0;
>      od:
>       for s from 0 to w-1 do
>        r:=0;
>        j:=j+1;
>         while(v[j]<=alpha or v[j]>beta) do
>          r:=r+1;
>          j:=j+1;
>         od:
```

```
>           if r>=t then
>            count[t]:=count[t]+1;
>           else
>            count[r]:=count[r]+1;
>           fi:
>       od:
>        q:=beta-alpha;
>         for k from 0 to t-1 do
>          p[k]:=q*(1-q)^k;
>         od:
>         p[t]:=(1-q)^t;
>         RETURN(evalf(1/w*sum(count[u]^2/p[u],
  u=0..t)-w));
> end:
>
```

4.3.2.3. EXEMPLOS

Vamos gerar uma seqüência de números pseudo-randômicos para os exemplos abaixo:

```
> _semente:=6097:
> for i from 0 to 10000 do
> Y[i]:=random();
> od:
>
>
```

Para uma seqüência de 200 números verificamos se as aberturas de no máximo tamanho 5 estão uniformemente distribuídas no intervalo [0 , 0.25].

```
> abertura(Y,200,5,0,1/4);
```

1.498600823

Para uma seqüência de 200 números verificamos se as aberturas de no máximo 5 graus de liberdade estão uniformemente distribuídas no intervalo [0.25 , 0.50].

```
> abertura(Y,200,5,1/4,1/2);
```

8.338353909

```
> abertura(Y,200,5,1/2,3/4);
```

3.298353909

```
> abertura(Y,2000,5,3/4,1);
```

8.908502058

Com os dados em mãos, agora basta analizarmos a tabela de valores χ^2 para verificar se a seqüência satizfaz a propriedade desejada.

O nosso programa retorna o valor do teste χ^2 sobre os dados obtidos através da observação das aberturas no intervalo [α , β].

```
>
```

4.3.3. TESTE SERIAL

4.3.3.1. CONCEITO

Pares de números inteiros devem estar uniformemente distribuídos, para verificar essa propriedade podemos dispor do teste serial.Tal teste verifica se os pares de números inteiros estão uniformemente distribuídos de maneira independente na seqüência de números pseudo-randômicos Y. Ou seja, os pares $(Y_{2j}, Y_{2j+1}) = (q, r)$, para $0 \le j$ e $j < n$ e $0 \le q$ e $r < d$, devem estar uniformemente distribuídos.

No teste, o grau de liberdade é $d^2 - 1$, pois existe essa quantidade de classes. Vejamos agora um programa que faz o teste.

4.3.3.2. PROGRAMA

Sintaxe:

serial (v, w, d)

onde *v* é o nome da seqüência
w é o comprimento da seqüência
d é utilizado neste programa para gerar uma nova seqüência Y_n de números para o qual o gerador trabalhará, o valor dessa nova seqüência é definida pelo valor do menor inteiro de *d* vezes o valor da seqüência aleatória V_n, que é gerada pela gerador em uso.

O grau de liberdade é k = $d^2 - 1$.

```
> serial:=proc(v::string, u::posint,d::posint)
>  local i,j,a,b,z,s,y,q,r,l,w;
>    w:=floor(u/2);
>    for i from 0 to w do
>     y[i]:=floor(d*v[i]);
>    od:
>     for j from 0 to d^2 do
>      s[j]:=0;
>     od;
>      i:=0;
>       for q from 0 to d-1 do
>        for r from 0 to d-1 do
>         for j from 0 to w-1 do
>          if y[2*j]=q and y[2*j+1]=r then
>           s[i]:=s[i]+1;
>          fi:
>         od:
```

```
>           i:=i+1;
>         od:
>       od:
>        z:=sum(s[o],o=0..d^2);
>        RETURN(evalf((1/z)*sum((s['l'])^2/(1/d^2),
  'l'=0..(d)^2)-z));
> end:
>
```

4.3.3.3. EXEMPLO

Vamos gerar a seqüência de números pseudo-randômicos U para os exemplos abaixo.

```
> _semente:=7864;
```

$$\_semente := 7864$$

```
> for i from 0 to 1000 do
> U[i]:=random();
> od:
```

Vamos testar a seqüência U para ver se quando ela for transformada em inteiros entre [0 , 2] ela apresente a propriedade desejada.

```
> serial(U,400,3);
```

$$7.820000000$$

Vejamos agora para números entre [0 , 3]

```
> serial(U,400,4);
```

$$12.96000000$$

E agora para números entre [0 , 5]

```
> serial(U,400,5);
```

$$23.$$

```
>
```

De posse dos dados, agora basta verificar a tabela de estatística χ^2 para averiguar se a seqüência testada apresenta a propriedade desejada. O programa apresentado também retorna, como o anterior, o valor da estatística χ^2 .

4.3.4. TESTE DE RODAGEM

4.3.4.1. CONCEITO

Podemos verificar se uma seqüência de números pseudo-randômicos possui segmentos de números crescentes e descrescentes uniformemente distribuídos num dado intervalo. Para isso pode-se testar essas seqüências com o teste de rodagem, que verifica quando uma seqüência tem tamanhos de segmentos decrescentes uniformes, rodagem para baixo, ou tamanhos de segmentos crescentes uniformes, rodagem para cima.Vejamos um exemplo:

Consideremos a seqüência de dez números dados a seguir:

1, 2, 9, 8, 5, 3, 6, 7, 0, 4

Agora vamos separar segmentos crescentes dessa seqüência:

|1, 2, 9 | 8 | 5 | 3, 6, 7 | 0, 4 |

Existe uma segmento ou rodagem de comprimento 3 seguido por uma rodagem de comprimento um, seguido por outra rodagem de comprimento um, seguido por uma rodagem de comprimento 3 e seguido por uma rodagem de comprimento 2.

Agora vamos separar segmentos decrescentes dessa seqüência:

| 1 | 2 | 9, 8, 5, 3 | 6 | 7, 0 | 4 |

Da mesma forma, existe uma rodagem de comprimento 1 seguido por outra de comprimento 1 seguido por outra de comprimento 4 seguido por outra de comprimento 1 seguido por outra de comprimento 2 seguido e finalmente, por outra de comprimento 1.

No teste de rodagem a estatística χ^2 não é utilizada. Nós utilizamos outra estatística baseada em dados retirados das observações e de dois conjuntos dados *a* e *b*. É examinado a tabela de valores de χ^2 para averiguar se as rodagens estão uniformemente distribuídas e a tabela é utilizada com seis e não cinco graus de liberdade, para o programa que foi desenvolvido.

Os valores estabelecidos para o conjunto de dados *a* e *b* são somente aproximados. É possível calcular os conjuntos de dados *a* e *b*, mas isso não será descrito aqui. O conjunto de dados *a* e *b* que é utilizado no programa do teste são para a estatística χ^2 com seis graus de liberdade, isto é $k = 7$.

4.3.4.2. PROGRAMA

A função ***rodagem*** foi definida para receber como primeiro argumentos o nome da seqüência a ser analizada, como segundo argumento o comprimento do intervalo e como terceiro argumento os comandos **baixo, cima** (se a rodagem é realizada para baixo ou para cima);

```
> rodagem:=proc(v::string,w::posint,z::string)
>  local j,i,r,a,b,count;
>    j:=-1;
>     for i from 0 to 6 do
>      count[i]:=0;
```

```
>     od:
>      while j<(w-1) do
>       r:=1;
>       j:=j+1;
>        if z=`cima` then
>         while  v[j]<v[j+1] and j<w do
>          r:=r+1;
>          j:=j+1;
>         od:
>        fi:
>         if z=`baixo` then
>          while  v[j]>v[j+1] and j<w do
>           r:=r+1;
>           j:=j+1;
>          od:
>         fi:
>          if r>=6 then
>           count[6]:=count[6]+1;
>          else
>           count[r]:=count[r]+1;
>          fi:
>      od:
>
> a:=matrix(6,6,[4529.4,9044.9,13568,18091,22615,27892,9044.
  9,18097,27139,36187,45234,55789,13568,27139,40721,54281,67
  852,83685,18091,36187,54281,72414,90470,111580,22615,45234
  ,67852,90470,113262,139476,27892,55789,83685,111580,139476
  ,172860]);
>
> b:=matrix(1,6,[1/6,5/24,11/120,19/720,29/5040,1/840]);
>
> print(V=1/w*evalf(sum(sum((count['q']-w*b[1,'q'])*(count['
  t']-w*b[1,'t'])*a['q','t'],'q'=1..6),'t'=1..6)));
> end:
>
```

4.3.4.3. EXEMPLOS

Vamos calcular a seqüência de números pseudo-randômicos para o teste:

```
> _semente:=89;
```

$$_semente := 89$$

```
> for i from 0 to 1000 do
> Y[i]:=random();
> od:
```

Vamos aplicar o teste de rodagem para uma seqüência de 500 números.

```
> rodagem(Y,500,baixo);
```

$V = 9.113124900$

```
> rodagem(Y,500,cima);
```

$V = 32.80372586$

Agora vamos aplicar o teste de rodagem para uma seqüência de 1000 números.

```
> rodagem(Y,1000,baixo);
```

$V = 12.97841666$

```
> rodagem(Y,1000,cima);
```

$V = 6.327971530$

```
>
```

4.3.6. TESTE DE PERMUTAÇÃO

4.3.6.1. CONCEITO

Uma seqüência de números pseudo-randômicos pode apresentar a propriedade de ter diferentes subseqüências de números sucessivos com diferentes ordenações. Gostaríamos que estas diferentes seqüências apresentassem uma ordenação uniforme e para isso, o teste de permutação é util.

O teste consiste em dividir a seqüência Y em n grupos de t elementos cada um e em cada grupo $t!$ ordenações. Fazemos então a observação dessas ordenações e então aplicamos o teste χ^2 com $k = t! - 1$ graus de liberdade e com probabilidade $\frac{1}{t!}$ para cada ordenação.

Para compreender melhor essa idéia de diferentes ordenações, vamos tomar um exemplo com $t = 3$, neste caso, nós teremos seis ordenações diferentes que são $(v_0 < v_1) < v_2$, ou $(v_0 < v_2) < v_1$,ou ... ou $(v_2 < v_1) < v_0$. Nós assumimos que a igualdade entre os valores de v não ocorre.

4.3.6.2. programa que avalia o valor máximo dentro de um intervalo [0.1]

Este programa determina onde está o maior número numa seqüência.

```
> maximo:=proc(x,i::integer,j::integer)
> local aux,mxim,l,pit;
> aux:=-10;
> for l from i to j do
>    if x[l]>aux then
>      aux:=x[l]:
>      pit:=l:
>    fi:
> od:
> RETURN(pit):
> end:
```

```
>
```

4.3.6.3. Teste que faz a análise

Este programa analiza uma permutação e retorna o valor correspondente a cada tipo de arranjo contida nela. Para ilustrar , veremos um exemplo com três números:

(a, b, c) = 5
(a, c, b) = 4
(b, a, c) = 2
(b, c, a) = 1
(c, a, b) = 0
(c, b, a) = 3

onde $a < b$ e $b < c$.

```
> analize:=proc(h,t::posint,ent::posint)
> local r,c,k,en,i,f,a,v:
> r:=t:
> en:=ent-1:
> for i from en*t to r+en*t-1 do
> v[i]:=h[i]:
> od:
> while r>0 do
> a:=maximo(v,en*t,r+en*t-1):
> c[r]:=(a-en*t):
> k:=v[r+en*t-1]:
> v[r+en*t-1]:=v[a]:
> v[a]:=k:
> r:=r-1:
> od:
> f:=sum(t!/(t-'i')!*c[t-'i'], 'i'=0..t-1):
> RETURN(f):
> end:
>
```

4.3.6.4. PROGRAMA

Este programa recebe *w* como sendo o valor do comprimento da seqüência e recebe *t* como sendo o valor de quantos números existem em cada subconjunto em que a seqüência será dividida. *v* é o nome da seqüência a ser testada.

```
> permutacao:=proc(v::string,w::posint,t::posint)
> local i,count,j,f,n;
> n:=floor(w/t):
> for i from 0 to t! do
```

```
> count[i]:=0;
> od:
> for i from 1 to n do
> f:=analize(v,t,i):
> count[f]:=count[f]+1:
> od:
> print(chi^2=evalf(1/n*sum(count['j']^2/(1/t!),'j'=0..t!-1)
  -n));
> end:
>
```

4.3.6.5. EXEMPLOS

Vamos gerar a seqüência de números pseudo-randômicos para o teste de randomicidade de permutação.

```
> _semente:=375:
> for i from 0 to 300 do
> v[i]:=random();
> od:
```

Vamos testar uma seqüência de 200 números com subseqüências contendo 3 números cada, onde cada uma não se sobrepõe.

```
> permutacao(v,199,3);
```

$$\chi^2 = 7.818181818$$

```
> permutacao(v,199,4);
```

$$\chi^2 = 13.20408163$$

```
>
```

Agora, basta analizar a tabela de valores χ^2 para verificar se a seqüência está satisfazendo a propriedade desejada. O nosso programa como os anteriores retorna o valor da estatística χ^2 já calculada.

4.3.7. TESTE DO MÁXIMO

Escolhemos subseqüências V da seqüência de números pseudo-randômicos U. Podemos formar uma seqüência constituída dos valores máximos das subseqüências V e então testar se essa nova seqüência está unifomemente distribuída.

Seja $V_j = \max(U_{tj}, U_{tj+1}, \ldots, U_{tj+t-1})$ para $0 \le j$ e $j < n$. Aplicamos o teste Kolmogorov-Smirnov para a seqüência $V_0, V_1, \ldots, V_{n-1}$ para verificar se está uniformemente distribuída. A função de distribuição é $F(x) = x^t$ onde $0 \le x$ e $x \le 1$. Podemos também aplicar o teste de equidistribuição para a seqüência $V_0^t, V_1^t, \ldots, V_{n-1}^t$ para verificar se a seqüência satisfaz as propriedade de randomicidade desejada.

Com os testes apresentados acima, nós concluímos a parte de testes para seqüências de números pseudo-randômicos, vale lembrar que o usuário pode e deve projetar seus próprios testes para as seqüências de números aleatórios.

Com os testes apresentados acima e com o conceito de números randômicos apresentado nos capítulos anteriores, poderemos trabalhar com algumas aplicações. Existem muitas áreas em que a randomicidade aparece, em especial na matemática e na estatística e em engenharias. Podemos usar geradores de números pseudo-randômicos para gerar as seqüências, para então trabalharmos com os problemas que aparecerem. Vejamos agora, alguns exemplos de problemas gerais, onde o uso de randomicidade pode ser útil para verificação de problemas e também para a sua resolução.